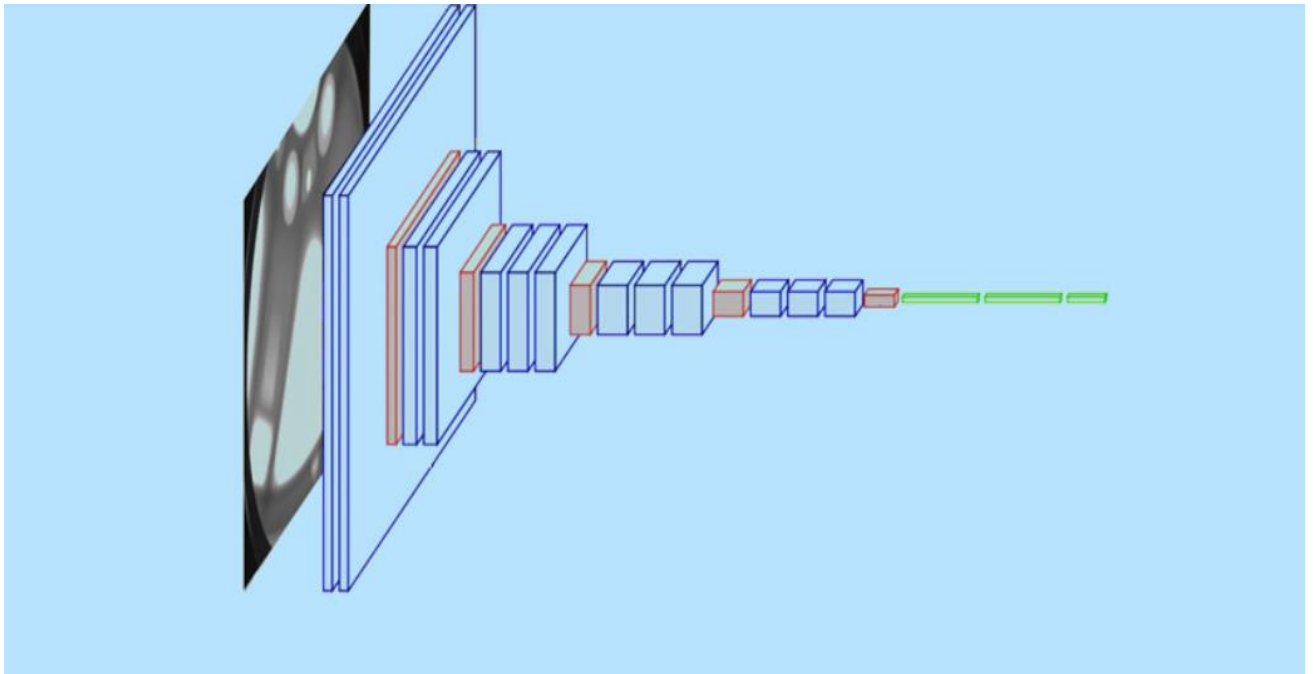




## Image scraping & Classification Project



Submitted by:  
SARMISHTHA HALDAR

## **ACKNOWLEDGEMENT:**

I would like to thank all my teachers, supervisors for the learning especially Shubham Yadav Sir. Few journals referred in the case are as follows:

- i) Expressive power of graph neural networks and the Weisfeiler-Lehman test by Michael Bronstein
- ii) Neural networks for pattern recognition Book by Christopher Bishop

## **Business Problem Framing**

Images are one of the major sources of data in the field of data science and AI. This field is making appropriate use of information that can be gathered through images by examining its features and details. We are trying to give you an exposure of how an end to end project is developed in this field.

The idea behind this project is to build a deep learning-based Image Classification model on images that will be scraped from e-commerce portal.

# INTRODUCTION

## Problem Statement

The idea behind this project is to build a deep learning-based Image Classification model on images that will be scraped from e-commerce portal after collecting the data from ecommerce portals

## Analytical Problem Framing

We have used methods like Accuracy for model evaluations

**Accuracy** is defined as the percentage of correct predictions

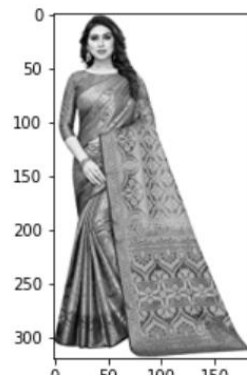
- Data Sources and their formats

We collected the data with the help of web scraping script which is present in github and then loaded the data in the environment as follows:

```
In [12]: import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
```

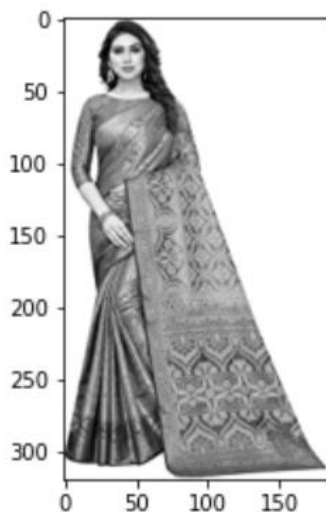
```
In [13]: DATADIR="C:/docs"
CATEGORIES=["Sarees", "Jeans", "Trousers"]

for category in CATEGORIES:
    path=os.path.join(DATADIR,category)
    for img in os.listdir(path):
        img_array=cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
        plt.imshow(img_array,cmap="gray")
        plt.show()
        break
    break
```



- Data Preprocessing

i)to resize the images

```
In [14]: #To resize the data to put all images to same size


```

ii) Creating training data

```
In [51]: #Create training data
```

```
In [62]: training_data=[]

def create_training_data():
    for category in CATEGORIES:
        path=os.path.join(DATADIR,category)
        class_num=CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                img_array=cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
                new_array=cv2.resize(img_array,(IMG_SIZE,IMG_SIZE))
                training_data.append([new_array,class_num])

            except Exception as e:
                pass

create_training_data()
```

```
In [16]: print(len(training_data))
```

5999

iii) To check the length of the training data to see if the dataset is balanced

```
create_training_data()

In [16]: print(len(training_data))
5999
```

```
In [54]: #Dataset is balanced
```

iv)To reshuffle the data to avoid overfitting

```
In [54]: #Dataset is balanced
```

```
In [55]: #Still Lets shuffle the data for better analysis
```

```
In [17]: import random
random.shuffle(training_data)
```

```
In [18]: for sample in training_data[:10]:
          print(sample[1])
```

```
2
2
2
2
0
2
0
0
0
0
1
```

- Hardware and Software Requirements , Tools Used  
No Specific requirements except Jupyter Notebook.

## Model/s Development and Evaluation

- Identification of possible problem

In this step, you will build the architecture of the classification convolutional neural network

- Run and Evaluate selected models: We ran and evaluated the above mentioned model

```
In [21]: from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D

In [33]: X=X/255.0

In [43]: y=np.array(y)

In [47]: model=Sequential()

In [48]: model.add(Conv2D(64, (3,3), input_shape=X.shape[1:]))
         model.add(Activation("relu"))
         model.add(MaxPooling2D(pool_size=(2,2)))

In [49]: model.add(Conv2D(64, (3,3)))
         model.add(Activation("relu"))
         model.add(MaxPooling2D(pool_size=(2,2)))

In [50]: model.add(Flatten())
         model.add(Dense(64))
         model.add(Activation("relu"))

In [51]: model.add(Dense(1))
         model.add(Activation('sigmoid'))

In [52]: model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=['accuracy'])

In [44]: model.fit(X,y, batch_size=32,validation_split=0.1)

169/169 [=====] - 80s 441ms/step - loss: -55757.6094 - accuracy: 0.3697 - val_loss: -402408.0312 - val_accuracy: 0.3767
```

```
In [79]: history = model.fit(X,y,epochs = 500 , validation_split=0.1)
```

Epoch 1/500

```
-----
Epoch 497/500
169/169 [=====] - 58s 341ms/step - loss: 0.0000e+00 - accuracy: 0.5253 - val_loss: 0.0000e+00 - val_accuracy: 0.4983
Epoch 498/500
169/169 [=====] - 65s 381ms/step - loss: 0.0000e+00 - accuracy: 0.5253 - val_loss: 0.0000e+00 - val_accuracy: 0.4983
Epoch 499/500
169/169 [=====] - 57s 338ms/step - loss: 0.0000e+00 - accuracy: 0.5253 - val_loss: 0.0000e+00 - val_accuracy: 0.4983
Epoch 500/500
169/169 [=====] - 57s 337ms/step - loss: 0.0000e+00 - accuracy: 0.5253 - val_loss: 0.0000e+00 - val_accuracy: 0.4983
```

- Key Metrics for success in solving problem under consideration

We found the accuracy

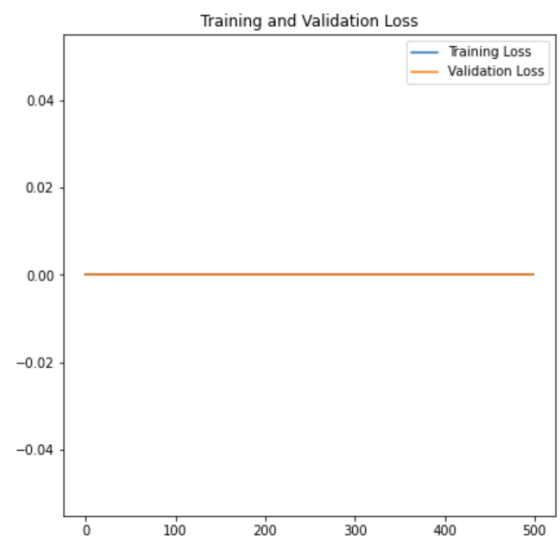
```
In [80]: acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(500)

plt.figure(figsize=(15, 15))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Training and Validation Accuracy

Training and Validation Loss



## CONCLUSION



- **Scope for Future**

The future of image processing will involve scanning the heavens for other intelligent life out in space. Also new intelligent, digital species created entirely by research scientists in various nations of the world will include advances in image processing applications. Due to advances in image processing and related technologies there will be millions and millions of robots in the world in a few decades time, transforming the way the world is managed. Advances in image processing and artificial intelligence<sup>6</sup> will involve spoken commands, anticipating the information requirements of governments, translating languages, recognizing and tracking people and things, diagnosing medical conditions, performing surgery, reprogramming defects in human DNA, and automatic driving all forms of transport. With increasing power and sophistication of modern computing, the concept of computation can go beyond the present limits and in future, image processing technology will advance and the visual system of man can be replicated. The future trend in remote sensing will be towards improved sensors that record the same scene in many spectral channels. Graphics data is becoming increasingly important in image processing applications. The future image processing applications of satellite based imaging ranges from planetary exploration to surveillance applications.

Using large scale homogeneous cellular arrays of simple circuits to perform image processing tasks and to demonstrate pattern-forming phenomena is an emerging topic. The cellular neural network is an implementable alternative to fully connected neural networks and has evolved into a paradigm for future imaging techniques. The usefulness of this technique has applications in the areas of silicon retina, pattern formation, etc